On Class Group Computations Using the Number Field Sieve

Mark L. Bauer[1] and Safuat Hamdy[2]

[1] University of Waterloo
Centre for Applied Cryptographic Research
Waterloo, Ontario, N2L 3G1
mbauer@math.uwaterloo.ca
[2] University of Calgary
Department of Mathematics and Statistics
Calgary, Alberta, T2N 1N4
hamdy@math.ucalgary.ca

Abstract. The best practical algorithm for class group computations in imaginary quadratic number fields (such as group structure, class number, discrete logarithm computations) is a variant of the quadratic sieve factoring algorithm. Paradoxical as it sounds, the principles of the number field sieve, in a strict sense, could not be applied to number field computations, yet. In this article we give an indication of the obstructions.

In particular, we first present fundamental core elements of a number field sieve for number field computations of which it is absolutely unknown how to design them in a useful way. Finally, we show that the existence of a number field sieve for number field computations with a running time asymptotics similar to that of the genuine number field sieve likely implies the existence of an algorithm for elliptic curve related computational problems with subexponential running time.

Keywords. imaginary quadratic number fields, class groups, number field sieve, imaginary quadratic function fields, hyperelliptic curve discrete logarithm.

1 Introduction

The best practical algorithm for class group computations in quadratic number fields so far is a variant of the quadratic sieve algorithm. In the imaginary quadratic case such computations include the computation of class structures, class numbers, discrete logarithms, and Diffie-Hellman secrets; in the real quadratic case such computations include the computation of regulators, fundamental units, and principal ideal generators. In this article we focus on the imaginary quadratic case, though, some arguments may be generalized to the real quadratic case or even to the case of number fields of arbitrary degree.

We refer to the quadratic sieve algorithm for the imaginary quadratic case as by the IQ-MPQS. The IQ-MPQS has an asymptotic running time proportional to $L_{|\Delta|}\left[\frac{1}{2}, c_1 + o(1)\right]$ for some positive constant c_1 (numerical evidence strongly suggests that $c_1 = 1$), where Δ is the discriminant.

1.1 Our Result

It is tempting to ask whether the number field sieve could not be used for number field computations as well. In fact, before the invention of the number field sieve the quadratic sieve was the best known algorithm to factor large integers, and the principles of the number field sieve could be profitably applied to many other computational problems which admit to algorithms of index-calculus type. However, paradoxical as it sounds, the number field sieve does not seem to work for number field computations.

In this article we give an indication of the obstructions in the imaginary quadratic case; we refer to the number field sieve in this case as by IQ-NFS. It must be clear, though, that if we ask for an IQ-NFS, then we mean to find an algorithm that is superior to the IQ-MPQS, i.e. having an asymptotic running time proportional to $L_{|\Delta|}\left[\frac{1}{3}, c_2 + o(1)\right]$ for some positive constant c_2, or even $L_{|\Delta|}\left[\frac{1}{2}, c_3 + o(1)\right]$ for some positive constant c_3 non-negligibly smaller than 1.

By examining the connection between the number field computations and function field computations, we also show that an IQ-NFS with a running time proportional to $L_{|\Delta|}\left[\frac{1}{3}, c_4 + o(1)\right]$ for some positive constant c_4 could almost certainly be exploited to develop an algorithm for elliptic curve related computational problems.

We will conclude that if there exists an IQ-NFS, it will most likely not be superior to the IQ-MPQS, and if it did, its design would probably not follow that of the genuine NFS.

We must point out that this article is of somewhat speculative nature, and thus it should be understood as a starting point for further research.

1.2 Cryptographic Relevance

We outline now briefly the cryptographic relevance of our results. There is a family of cryptographic public-key schemes based on the intractability of some computational problems with class groups of imaginary quadratic number fields [8]; we call these cryptographic schemes IQ-schemes. Due to the sparseness of independent computational problems that admit to efficient cryptographic schemes, these public-key schemes were introduced as an alternative to existing schemes. More precisely: the security of the cryptographic schemes that are used in practice is based on the intractability of very few families of independent computational problems. Moreover, rigorous and unconditional proofs of the intractability of any of those computational problems are not known. This has repeatedly raised concerns about public-key cryptography. It is therefore advisable to have some well worked out and independent alternatives available. IQ-cryptography provides such an alternative; IQ-schemes are secure (using standard definitions and models of security) and efficient (in a practical sense).

The best known algorithm to solve these computational problems is, as already mentioned, the IQ-MPQS with the running time asymptotic $L\left[\frac{1}{2}\right]$. Traditional cryptographic schemes are based on the intractability of factoring integers or finite field computations, and the best known algorithms to solve these computational problems are variants of the number field sieve with the running time asymptotic $L\left[\frac{1}{3}\right]$. Thus, it seems that there is a complexity theoretic gap between IQ-related computational problems and factoring or finite field related problems.

Such a gap implies that, with increasing security level, the sizes of cryptographic parameters (such as RSA moduli, finite field size, discriminants etc.) and thus operands in a cryptographic operation (such as computing a signature) grow faster for traditional schemes than for IQ schemes. In spite of the more complex IQ-arithmetic it follows that IQ-cryptography eventually outperforms traditional cryptography. It is clear, though, that IQ-cryptography is eventually inferior to elliptic curve cryptography for the same reason. Yet, for the time being, IQ cryptography can be, in principle at least, considered as an efficient alternative.

However, the main motivation of IQ-cryptography is not its efficiency. We finally mention that due to the fact that the orders of class groups are in general not efficiently computable, IQ-cryptography has applications where elliptic curves do not work, see e.g. [5].

1.3 Notation

We use the following common notation

$$L_x[\varepsilon, c] = \exp\left(c(\log x)^{\varepsilon}(\log\log x)^{1-\varepsilon}\right) \quad ,$$

while $L[\varepsilon]$ is the abbreviation for $L_x[\varepsilon, c]$ for some variable x and some positive constant c.

2 Constructive Obstructions

In this section we present some obstructions one encounters if one wants to design an IQ-NFS along the lines of the genuine NFS.

2.1 A Brief Review of the NFS Relation Generation

We begin with a brief review of the relevant details of the NFS relation generation for DL computations in finite fields, see [7, 12–15] as well as [2] for all details. Let p be a prime and let $\mathbb{F}_p$ be a finite field. Then let d be a suitably chosen (small) integer, take $m = \lfloor p^{1/d} \rfloor$, and for $0 \leqslant i \leqslant d$ let a_i be the digits of the base-m expansion of p, i.e. let a_i be non-zero integers such that $p = \sum_{0 \leqslant i \leqslant d} a_i m^i$. Then let $f(X)$ be the polynomial with coefficients a_i of degree d. Suppose that $f(X)$ is irreducible over $\mathbb{Z}$, and let α denote a root of $f(X)$. Since

$$f(m) \equiv 0 \pmod{p} \tag{1}$$

the map defined by $\phi(\alpha) \mapsto m$, is a ring homomorphism from $\mathbb{Z}[\alpha]$ to $\mathbb{F}_p$. Formally one is looking at $\mathbb{Z}[\alpha]$-integers of the form $\theta = a - b\alpha$, of which the norm is $\mathrm{N}(a - b\alpha) = F(a, b)$, where $F(X, Y)$ is the homogenized bivariate polynomial that corresponds to $f(X)$. A $\mathbb{Z}[\alpha]$-integer θ is understood to be smooth if its norm is smooth. The main task in the sieving stage is to find a set of coprime integers a and b such that θ and $\phi(\theta)$ are simultaneously smooth. That is, $F(a, b)$ and $a - bm$ have to be smooth simultaneously. This is done by taking $G(X, Y) = F(X, Y)(X - mY)$ and sieving the bivariate polynomial $G(X, Y)$.

Based on the bounds on X, Y and the coefficients of G, and assuming that the values of $G(X, Y)$ behave like random integers with respect to smoothness probability, one gets that the running time is proportional to $L_p\left[\frac{1}{3}, c_4 + o(1)\right]$, where $c_4 = (64/9)^{1/3}$; moreover, $d = \left\lfloor (3 \ln p / \ln \ln p)^{1/3} \right\rfloor$.

In order to get the favorable running time for the NFS the following items are crucial:

1. The degree d of f (and thus of G) tends uniformly to infinity as p tends to infinity.
2. The size of the coefficients of G are of order $p^{1/d}$.
3. There is an efficient way to select a polynomial and thus an extension of $\mathbb{Q}$.
4. There is an efficiently computable homomorphism from $\mathbb{Z}[\alpha]$ to $\mathbb{F}_p$.
5. The sieving is done in the two domains $\mathbb{Z}[\alpha]$ and $\mathbb{F}_p$ simultaneously.

We will outline below that it is unknown how to achieve any of these items in the number field case.

2.2 A Brief Review of the IQ-MPQS Relation Generation

Before we proceed we shall briefly review the IQ-MPQS relation generation, see [9, 10] for details. Let $\mathcal{O}_\Delta$ denotes the quadratic order of discriminant Δ. The objective is to find a set $\mathcal{R}$ of *relations* R_i of the form

$$R_i : \prod_{\mathfrak{p}_j \in \mathcal{FB}} \mathfrak{p}_j^{e_{i,j}} \sim \mathcal{O}_\Delta \ . \tag{2}$$

Here $\mathcal{FB}$ is the *factor base*, a set of primitive prime $\mathcal{O}_\Delta$-ideals of the form (p, b) where $p \leqslant B$ for some bound B. For each prime ideal $\mathfrak{p}_j = (p_j, b_j)$ of $\mathcal{FB}$, let $b_j \geqslant 0$; the prime ideal $\bar{\mathfrak{p}}_j = (p_j, -b_j)$ will be represented by $\mathfrak{p}_j^{-1}$.

In order to generate a relation, a $\mathcal{FB}$-smooth $\mathcal{O}_\Delta$-ideal is constructed. Let $\mathfrak{a}$ be this ideal with the representation (a, b). The corresponding binary quadratic form is $A(X, Y) = aX^2 + bXY + cY^2$, where $c = (b^2 - \Delta)/4a$. Now, if there are coprime integers x and y such that $ax^2 + bxy + cy^2 = a'$, then there exists another binary quadratic form $A'(X, Y) = a'X^2 + b'XY + c'Y^2$, which is equivalent to A, and in fact, the corresponding $\mathcal{O}_\Delta$-ideal $\mathfrak{a}' = (a', b')$ is equivalent to $\mathfrak{a}$. (The integer b' can be efficiently computed from the integers a, b, c, x, and y.) Therefore, $\mathfrak{a}\mathfrak{a}'^{-1} \sim \mathcal{O}_\Delta$, and if $\mathfrak{a}'$ is $\mathcal{FB}$-smooth, this constitutes a relation. The prime ideal factorization of $\mathfrak{a}'$ can be obtained from the prime factorization of a', and from b' and Δ.

In order to find x and y such that $A(x, y) = a'$ is smooth, we sieve the quadratic polynomial $A(X, Y)$; for simplicity, fix $Y = 1$. Then the sieving is performed almost exactly as in the MPQS factoring algorithm. Likewise, the selection of polynomials is exactly as in the MPQS factoring algorithm, including the self initialization technique.

The sieving step of the IQ-MPQS is, in a remote sense, similar to its counterpart in the NFS. In both algorithms polynomials are sieved. Yet, none of the crucial properties above are satisfied. In particular, the degree of the sieving polynomials is fixed, no matter how large Δ is, the size of the coefficients of the quadratic polynomials are of the order of $|\Delta|^{1/2}$, and the sieving takes place only in one domain.

2.3 Towards an IQ-NFS

In this section we try to build an IQ-NFS on top of the IQ-MPQS. We proceed rather naively and follow along the lines of the genuine NFS.

Finding a Suitable Extension and a Homomorphism First we try to find a suitable extension of $\mathcal{O}_\Delta$. In the genuine NFS there was a natural way to find an irreducible polynomial over $\mathbb{Z}$: we took p and from it computed an integer m and coefficients of a polynomial $f(x)$ such that $f(m) \equiv 0 \pmod{p}$; in particular, $f(m) = p$, see above. Now, p was the characteristic of the finite field, which is a prime. However, in the number field case, the characteristic is always 0. So, it remains to be seen what to put in the place of p in the number field case.

Now, recall that the procedure in the genuine NFS to find a polynomial is not only natural because it is very simple and efficient. More important, we get the necessary ring-homomorphism from $\mathbb{Z}[\alpha]$ to $\mathbb{F}_p$, and this homomorphism is very efficient to compute. It is this very connection between the polynomial and the homomorphism that makes, for instance, the difference between the generalized number field sieve (with rather large polynomial coefficients) and the special number field sieve (with very small polynomial coefficients). If the coefficients in the GNFS could be chosen freely, then there would be no difference between the GNFS and the SNFS. However, it is not known how to find a suitable homomorphism for arbitrary polynomials, and therefore, the polynomial must be chosen as described above.

The same is certainly true in the number field case, where we have the additional problem of what to put in the place of p. We note, though, that in the number field case we are interested in a group-homomorphism instead of a ring-homomorphism. For instance, let $K = \mathbb{Q}(\sqrt{\Delta})$ and let $\mathcal{O}_K$ be the maximal order of K; likewise, let L be an extension of K and let $\mathcal{O}_L$ be the maximal order of L.

What we are looking for is a group homomorphism ψ that maps $\mathcal{O}_L$ to $\mathcal{O}_K$. Since a (basic) IQ-NFS algorithm would search for pairs $(\mathfrak{A}, \mathfrak{a})$, where $\mathfrak{A}$ is an $\mathcal{O}_L$-ideal and $\mathfrak{a}$ is an $\mathcal{O}_K$-ideal, ψ must satisfy the following properties:

1. if $\mathfrak{A}$ is smooth (in a suitable sense), then $\psi(\mathfrak{A})$ is smooth;

2. $\psi(\mathfrak{A}) \sim \mathfrak{a}$;
3. ψ is efficiently computable.

In the face of the fact that $\mathfrak{A}$ and $\mathfrak{a}$ must be found *simultaneously* (by sieving a polynomial), property 2. in conjunction with property 3. appear to be the hardest to satisfy. In fact, it is unknown how one could do that.

Finding a Polynomial with Small Coefficients Suppose for the moment that we have surmounted the obstructions from the previous subsection. It is tempting to ask what one would get out of the algorithm, i.e. what would be its asymptotic expected running time. Following the design of the genuine NFS, we proceed naively in the following way:

1. Choose a suitable integer d.
2. Choose an extension over K, i.e. choose a polynomial $f(X) \in \mathcal{O}_K[X]$ such that
$$f(X) = \alpha_d X^d + \alpha_{d-1} X^{d-1} + \cdots + \alpha_0 \ , \tag{3}$$
where $\alpha_i = a_i + b_i\sqrt{\Delta}$. Let $|a_i|, |b_i| \leqslant B$, where B is a bound, e.g. $B = |\Delta|^{1/d}$ (recall that we just proceed naively as in the genuine NFS).
3. For the sieving we need a polynomial over $\mathbb{Z}$. In order to get such a polynomial $f_{\mathbb{Z}}(X)$ from $f(X)$, rewrite $f(X) = a(X) + b(X)\sqrt{\Delta}$ and let $\bar{f}(X) = a(X) - b(X)\sqrt{\Delta}$ be the conjugate polynomial. Now let $f_{\mathbb{Z}}(X) = f(X)\bar{f}(X) = a^2(X) - b^2(X)\Delta$. Note that since we are dealing with imaginary quadratic number fields, $\Delta < 0$ and thus $f_{\mathbb{Z}}(X) = a^2(X) + b^2(X)|\Delta|$. Now it becomes apparent that $f_{\mathbb{Z}}(X)$ has coefficients of the order of $|\Delta|^{1+2/d}$, which turns out to be too large in order to get the $L\left[\frac{1}{3}\right]$ running time asymptotics, see below.
4. Finally, let $F_{\mathbb{Z}}(X,Y)$ be the homogenized form of $f_{\mathbb{Z}}(X)$. (We presume that a smooth value for $F_{\mathbb{Z}}(X,Y)$ would in some way give rise to a smooth $\mathcal{O}_L$-ideal.) Let $A(X,Y)$ be the binary quadratic form that corresponds to an $\mathcal{O}_K$-ideal $\mathfrak{a}$, take $G(X,Y) = F_{\mathbb{Z}}(X,Y)A(X,Y)$ and sieve $G(X,Y)$ for pairs (x,y) such that $G(x,y)$ is smooth. Since the coefficients of $F_{\mathbb{Z}}$ are of size $O(|\Delta|B^2) = O(|\Delta|^{1+2/d})$ and the coefficients of A are of size $O(|\Delta|^{1/2})$, the coefficients of G are of size $O(|\Delta|^{3/2+2/d})$.

For the running time analysis we use the following principle from [2, Section 10]: Let $L(Z) = \exp\left(\sqrt{\ln Z \ln\ln Z}\right)$. In a sequence of $L(Z)^{\sqrt{2}+o(1)}$ random integers uniformly chosen from the interval $[0,Z]$ $S = L(Z)^{1/\sqrt{2}+o(1)}$ of them will be S-smooth, and this is the optimal choice for S in order to maximize the yield.

We apply this principle to the sequence of integers that we get from $G(X,Y)$ for X and Y ranging over certain intervals; here we assume that the integers $G(X,Y)$ have the same properties as ordinary integers with respect to smoothness-probability (this constitutes, as usual, the major heuristic leap in the running time analysis).

We have $Z = |G(X,Y)|$, and since we sieve two-dimensionally, as in the genuine NFS, we have $|X|, |Y| \leqslant M$ where $M = L(Z)^{1/\sqrt{2}+o(1)}$. Now we are in the position to perform a running time analysis as in [2], see also [4, Section 6.2.3].

- If d is *fixed* as $|\Delta| \to \infty$, then we get an asymptotic running time proportional to $L_{|\Delta|}\left[\frac{1}{2}, 3 + \frac{4}{d} + o(1)\right]$.
- If $d \to \infty$ as $|\Delta| \to \infty$, then we get an asymptotic running time proportional to $L_{|\Delta|}\left[\frac{1}{2}, \sqrt{3} + o(1)\right]$.

This means that the IQ-NFS (designed as above) performs in any case much worse than the IQ-MPQS. Now, $f_{\mathbb{Z}}$ as chosen above may not be the optimal polynomial for L. One could, for example, use lattice reduction methods to get a polynomial $\overline{f_{\mathbb{Z}}}$ with smaller coefficients, see for example Algorithm POLRED in [3, Algorithm 4.4.1]. However, the coefficients of such a polynomial will not be arbitrary small, and even if $f_{\mathbb{Z}}$ had coefficients of order $O(|\Delta|^{1/2+2/d})$ (which constitutes a substantial improvement), then the asymptotic running times got merely down to $L_{|\Delta|}\left[\frac{1}{2}, 2 + \frac{4}{d} + o(1)\right]$ and $L_{|\Delta|}\left[\frac{1}{2}, \sqrt{2} + o(1)\right]$, which is still worse than the IQ-MPQS. We will elaborate the effectiveness of polynomial reduction algorithms applied to our problem in the full version of the paper. Finally, changing the polynomial also changes the basis for element and ideal representation in L, and thus, this changes the homomorphism; that might be a major problem.

In order to get the typical NFS asymptotic $L\left[\frac{1}{3}\right]$, the coefficients of $G(X, Y)$ must have order of magnitude $|\Delta|^{O(1/d)}$. Since the coefficients of $A(X, Y)$ usually have order of magnitude $|\Delta|^{1/2}$ we must alter the design of the IQ-NFS. Let $F_{\mathbb{Z},1}(X, Y)$ and $F_{\mathbb{Z},2}(X, Y)$ irreducible polynomials with the desired properties, let L_1 and L_2 be the corresponding extension fields, let $G(X, Y) = F_{\mathbb{Z},1}(X, Y)F_{\mathbb{Z},2}(X, Y)$, and let ψ_1 and ψ_2 be ideal-homomorphisms that map $\mathcal{O}_{L_1}$-ideals and $\mathcal{O}_{L_2}$-ideals to $\mathcal{O}_K$-ideals. Now we require that if a smooth $\mathcal{O}_{L_1}$-ideal $\mathfrak{A}_1$ and a smooth $\mathcal{O}_{L_2}$-ideal $\mathfrak{A}_2$ are found simultaneously by sieving $G(X, Y)$, then $\psi_1(\mathfrak{A}_1)$ and $\psi_2(\mathfrak{A}_2)$ are also smooth and $\psi_1(\mathfrak{A}_1) \sim \psi_2(\mathfrak{A}_2)$. Still, it remains to be seen how $F_{\mathbb{Z},1}$ and $F_{\mathbb{Z},2}$ as well as $\psi_1(\mathfrak{A}_1)$ and $\psi_2(\mathfrak{A}_2)$ are to be chosen.

Summary The major stumbling blocks on the way towards an IQ-NFS are firstly to find suitable extensions of imaginary quadratic number fields, which provide suitable ideal-homomorphisms. It is unknown how to find such extensions. Secondly, by the nature of sieving algorithms, the extensions are to be represented as irreducible polynomials over $\mathbb{Z}$. It is unknown how to find suitable irreducible polynomials with sufficiently small coefficients. The first obstruction says that it is not known how to design core elements of the IQ-NFS, and the second obstruction says that even so it is still unknown how the IQ-NFS will be of any use.

3 Relative Obstructions

In this section we will attempt to provide a connection between the aforementioned problems and those that arise in the case of elliptic curves and hyperelliptic curves. We begin by stating the following definition.

Definition 1. *The* Discrete Logarithm Problem *in $\mathcal{G}$ is: given $\gamma, \gamma' \in \mathcal{G}$, find the smallest $n \in \mathbb{Z}_{>0}$ such that $\gamma^n = \gamma'$ if such an integer exists.*

If the group under consideration corresponds to the points on an elliptic curve, we call this the *Elliptic Curve Discrete Logarithm Problem*, and abbreviate it by ECDLP. Analogously, if the groups corresponds to the Jacobian of a hyperelliptic curve, we call this the *Hyperelliptic Curve Discrete Logarithm Problem* and denote it by HCDLP. Since an elliptic curve is just a hyperelliptic curve of genus one, we note that the ECDLP is just a particular instance of the HCDLP.

The majority of this section will describe how to take an instance of the ECDLP and convert it to an instance of an HCDLP for a curve of higher genus. That is, we will prescribe a technique for constructing a cover of an elliptic curve by a hyperelliptic curve of larger genus that forces an inclusion from the elliptic curve into the Jacobian of the hyperelliptic curve. While this is apparently a well known result, we include some details since they appear to be lacking from the literature.

We conclude the section by discussing how the existence of this map relates to the overall complexity of solving the ECDLP. We will also discuss how finding an algorithm of lower subexponential complexity for the HCDLP seems intrinsically linked to solving the analogous problem for imaginary quadratic number fields.

3.1 Jacobians of Hyperelliptic (and Elliptic) Curves

In the remaining sections, let $\mathbf{K}$ denote an arbitrary field. We will mostly be interested in the case when $\mathbf{K} = \mathbb{F}_q$, but the majority of what follows applies to arbitrary fields. For our purposes, it suffices to define a hyperelliptic curve of genus g to be a curve given by an equation of the following form:

$$C : y^2 + h(x)y = f(x)$$

where $h, f \in \mathbf{K}[x]$ are such that $\deg h \leqslant g$ and $\deg f = 2g + 1$ or $2g + 2$ with f monic. Furthermore, no element in $\overline{\mathbf{K}} \times \overline{\mathbf{K}}$ may simultaneously satisfy

$$y^2 + hy - f = 0 \ , \quad 2y + h = 0 \ , \quad h'y - f' = 0 \ .$$

These last criteria force the curve to have a smooth affine model, which simply makes calculations more palatable (and the statement about the genus correct). Every hyperelliptic curve inherently admits such a model, so this by no means limits our discussion. Furthermore, if the characteristic of $\mathbf{K}$ is not 2, we will always take $h = 0$ (this is possible by completing the square on the left hand side). A hyperelliptic curve of genus one is an elliptic curve. The *function field* of C is defined to be

$$\overline{\mathbf{K}}(C) \cong \overline{\mathbf{K}}(x)[y]/(y^2 + hy - f) \ .$$

Each element of the function field can be thought of as a map from C to $\overline{\mathbf{K}} \cup \{\infty\}$ (otherwise denoted as $\mathbb{P}^1_{\overline{\mathbf{K}}}$).

We now give a brief overview of the Jacobian of a hyperelliptic curve. For more complete details, see the appendix in [11]. A *divisor* on C is a formal sum of points $D = \sum m_P P$ where $m_P = 0$ for all but finitely many points of C. The degree of a divisor is $\deg D = \sum m_P$. The set of all divisors on C forms a group and is denoted $\mathrm{Div}(C)$. The subset of all divisors of degree zero is a proper subgroup and is denoted $\mathrm{Div}^0(C)$.

We wish to consider the quotient of $\mathrm{Div}^0(C)$ by the following subgroup. For any function $\gamma \in \overline{\mathbf{K}}(C)$, we may associate a divisor to γ by $(\gamma) = \sum m_P P$ where $m_p = \mathrm{ord}_P(\gamma)$ is the order of the zero or pole of γ at P. Such divisors are said to be *principal divisors*. The set of all principal divisors is denoted by $\mathcal{P}(C)$. $\mathcal{P}(C)$ is a subgroup of $\mathrm{Div}^0(C)$ because every principal divisor has degree zero, although this is by no means obvious from the above definitions.

The group that we are interested in is called the *Picard group* of C (in fact, we are interested in the degree zero part of the Picard group, but we will abuse the language slightly). The group is defined to be

$$\mathrm{Pic}^0(C) \cong \mathrm{Div}^0(C)/\mathcal{P}(C) \ .$$

$\mathrm{Pic}^0(C)$ contains all of the arithmetic information about the Jacobian of C that we need. If we have a tower of fields, $\mathbf{K} \subseteq L \subseteq \overline{\mathbf{K}}$, then $G_L = \mathrm{Gal}(\overline{\mathbf{K}}, K)$ has a natural action on $\mathrm{Pic}^0(C)$ induced by its action on $\mathcal{P}(C)$ and $\mathrm{Div}^0(C)$ (which conveniently agree). The fixed group under this action is denoted by $\mathrm{Pic}^0_L(C)$. Obviously, we will be most interested in the case when $L = \mathbf{K}$.

While the above construction of $\mathrm{Pic}^0(C)$ is mathematically rigorous, it is somewhat lacking from a computational perspective. We will not cover the details of how to perform arithmetic, but instead refer the reader to the appendix in [11] again. The second computational problem that arises is how to represent elements in this group. For each element in $\mathrm{Pic}^0(C)$, it is possible to associate to it a unique divisor in $\mathrm{Div}^0(C)$. These unique divisors are called *reduced divisors.* The arithmetic and presentation in $\mathrm{Pic}^0(C)$ are performed using these reduced divisors.

3.2 Including Elliptic Curves into Jacobians of Hyperelliptic Curves

We begin with a definition to facilitate in constructing the desired cover of our elliptic curve. Let p denote the characteristic of the field $\mathbf{K}$. For an integer n, we will write $n = n_1 \cdot p^{n_p}$ where n_1 is an integer that satisfies $\gcd(n_1, p) = 1$, and $n_p \geqslant 0$ (in characteristic zero, one takes $n_1 = n$, $n_p = 0$). Let $\mathfrak{p}_p(x)$ denote the Artin-Schreier character, i.e. $\mathfrak{p}_p(x) = x^p - x$. Define the polynomial

$$\mathfrak{C}_n(x) = \mathfrak{p}_p(x)^{\circ n_p} \circ x^{n_1} \ .$$

Theorem 1. *Let E be an elliptic curve given by*

$$E : y^2 + hy = f \;\;, \quad \deg f = 3 \;\;, \quad \deg h \leqslant 1 \;\;.$$

If in characteristic 2, $h(0) \neq 0$ or in characteristic different from 2, $f(0) \neq 0$, then there exists a hyperelliptic curve C_n of genus $\lfloor n + \frac{n-1}{2} \rfloor$ given by

$$C_n : y^2 + h(\mathfrak{C}_n(x))y = f(\mathfrak{C}_n(x)) \;\;,$$

such that C_n is an n-to-1 cover of E.

The restrictions on $h(0)$ and $f(0)$ ensure that the resulting model for C_n is smooth. The smoothness of this model is a consequence of combining the definition of smoothness given above and noting that the polynomial given by $\mathfrak{C}_n(x) - \alpha$ for any $0 \neq \alpha \in \overline{\mathbf{K}}$ has no repeated roots. The genus follows trivially from the definition given above for a hyperelliptic curve.

Considering an elliptic curve over $\mathbf{K}$, it is possible to transform it into a curve of this form by using the substitution $x \mapsto x + \alpha$ for some $\alpha \in \mathbf{K}$ satisfying $h(\alpha) \neq 0$ in characteristic 2, or $f(\alpha) \neq 0$ otherwise. The only elliptic curve for which finding such an α is not possible is the curve

$$E : y^3 = x(x-1)(x-2)$$

defined over $\mathbb{F}_3$. By extending the ground field, the above substitution could then be used. However, this curve is of no interest for the problem we wish to solve, so the above theorem applies to all cryptographically interesting elliptic curves.

The map from C_n to E is given as follows.

$$\Psi_n : C_n \to E$$

$$(\alpha, \beta) \mapsto (\mathfrak{C}_n(\alpha), \beta) \;\;,$$

and the point(s) at infinity on C_n map to the unique point at infinity on E. This is clearly a well-defined algebraic map of curves, and for most points, there are precisely n distinct pre-images under Ψ since $\mathfrak{C}_n(x) - \alpha$ has degree n. Therefore, C_n is an n-to-1 cover of E via Ψ_n.

Given any two curves and a map between them, there is an induced map on the Jacobians of the two curves. We can use the map Ψ_n defined above to do precisely this. We proceed by constructing a map between the respective divisor class groups

$$\Psi_n^* : \mathrm{Div}^0(E) \to \mathrm{Div}^0(C_n) \;\;.$$

We define the map as follows. Let $P = (\alpha, \beta)$ be a finite point on E, and let α_i denote the n roots of $\mathfrak{C}_n(x) - \alpha$ (each with appropriate multiplicity). If n is odd, then

$$\Psi_n^* : P - P_\infty \to \left(\sum_{i=1}^{n} (\alpha_i, \beta) \right) - nP_\infty$$

where P_∞ represents the unique point at infinity on the two respective curves. If n is even,

$$\Psi_n^* : P - P_\infty \to \left(\sum_{i=1}^{n} (\alpha_i, \beta) \right) - \frac{n}{2} \left(P_{\infty_1} + P_{\infty_2} \right)$$

where P_∞ is the unique point at infinity on E, and P_{∞_1} and P_{∞_2} are the two points at infinity on C_n. Since Ψ_n^* includes $\mathcal{P}(E)$ into $\mathcal{P}(C)$, it induces a map on the Picard groups, called the *conorm* map:

$$\mathrm{Con}_{C_n/E} : \mathrm{Pic}^0(E) \to \mathrm{Pic}^0(C_n) \enspace .$$

It is precisely this map that we will use to translate an ECDLP into an HCDLP for a curve of higher genus.

Theorem 2. *If E and C_n are as specified in theorem 1, then the induced map*

$$\mathrm{Con}_{C_n/E} : \mathrm{Pic}^0_L(E) \to \mathrm{Pic}^0_L(C_n)$$

is injective for all n and any $\mathbf{K} \subseteq L \subseteq \overline{\mathbf{K}}$.

By first proving the result over the algebraic closure, $\overline{\mathbf{K}}$, the theorem follows for all intermediary subfields by restriction. In our case, we are really only interested in the case of $L = \mathbf{K}$. For the case when n is odd, it is simple enough to show that the image under Ψ_n^* of a divisor on E of the form $P - P_\infty$, where P is any finite point, is a non-trivial reduced divisor in $\mathrm{Div}^0(C_n)$. This is sufficient to conclude that the map in injective. If n is even, one proves the injectivity of the map by constructing a secondary hyperelliptic curve which is isomorphic to C_n, but has only one point at infinity.

Since this map is injective, it is clear that given an ECDLP for E, we can translate it into an HCDLP for C_n. This map is effective and quite easy to compute using $\mathfrak{C}_n$. If n is odd and we are using the standard representations for divisors on C_n, the map is given by

$$(\alpha, \beta) \mapsto \mathrm{div}(\mathfrak{C}_n(x) - \alpha, \beta) \enspace .$$

3.3 Relating the Complexity of the ECDLP and HCDLP

Although C_n can be used to convert an ECDLP into an HCDLP, this does not necessarily help us solve the problem more efficiently. In fact, with the current algorithms for solving instances of HCDLP's, this amounts to taking a hard problem and making it harder. However, in this section, we consider the ramifications of the development of an algorithm to solve the HCDLP that is considerably more efficient than the algorithms that currently exist.

We start by noting that for a hyperelliptic curve of genus g over $\mathbb{F}_q$, the size of $\mathrm{Pic}^0_{\mathbb{F}_q}(C)$ is roughly q^g.

Theorem 3. *If there exists an algorithm to solve the HCDLP with running time $L_{q^g}[\alpha, \beta + o(1)]$ with $\alpha < 1/2$ for $g \approx \log q$, as $q \to \infty$, then there exists an algorithm to solve the ECDLP in time $L_q[\alpha', \beta' + o(1)]$ with $\alpha' < 1$ and $\beta' \geqslant 0$ as $q \to \infty$.*

Proof. Given an elliptic curve E over $\mathbb{F}_q$, set $n = \left\lceil \frac{2}{3} \log q \right\rceil$. By using $\mathrm{Con}_{C_n/E}$ and C_n which has genus $g \geqslant \log q$, we can solve the HCDLP in $\mathrm{Pic}^0(C_n)$ in time $L_{q^g}[\alpha, \beta + o(1)]$. Letting γ be such that $g = \gamma \log q$, then we have

$$(\beta + o(1))(\log q^g)^\alpha (\log\log q^g)^{1-\alpha} = (\beta + o(1))(\log q)^{2\alpha} \gamma^\alpha (\log \gamma + 2 \log\log q)^{1-\alpha} \ .$$

Ignoring the coefficients for a moment, we may rewrite the right hand side as

$$(\log q)^{2\alpha+\epsilon} (\log\log q)^{1-(2\alpha+\epsilon)} \frac{(\log \gamma + 2\log\log q)^{1-\alpha}}{(\log\log q)^{1-(2\alpha+\epsilon)} (\log q)^\epsilon}$$

for any $\epsilon > 0$. As $q \to \infty$, the fractional term tends to 0 (since $\gamma \to 1$), and hence we have that

$$(\log q^g)^\alpha (\log\log q^g)^{1-\alpha} \gg (\log q)^{2\alpha+\epsilon} (\log\log q)^{1-(2\alpha+\epsilon)} \ .$$

Therefore, an upperbound for the running time is given by $L_q[2\alpha + \epsilon, \beta']$, for any positive fixed $\epsilon > 0$, and any $\beta' \geqslant 0$ (although clearly both affect the constants involved in the big O-notation). If $\alpha < 1/2$, we can clearly choose $\epsilon > 0$ such that $2\alpha + \epsilon < 1$, which proves the desired result.

Theorem 4. *If there exists an algorithm to solve the HCDLP in time $L_{q^g}[\alpha, \beta + o(1)]$ with $\alpha = 1/2$ for $g \approx (\log q)^\delta$ and $\delta < 1$, as $q \to \infty$, then there exists an algorithm to solve the ECDLP in time $L_q[\alpha', \beta' + o(1)]$ with $\alpha' < 1$ and $\beta' \geqslant 0$ as $q \to \infty$.*

Proof. We proceed as above, but this time we choose $n = \left\lceil \frac{3}{2} (\log q)^\delta \right\rceil$. As before, we map to $\mathrm{Pic}^0(C_n)$, where we can solve the HCDLP in time $L_{q^g}[\alpha, \beta + o(1)]$. Letting γ be such that $g = \gamma (\log q)^\delta$, then after substituting for g we note

$$(\beta + o(1)) (\log q^g)^\alpha (\log\log q^g)^{1-\alpha} = (\beta + o(1)) (\log q)^{(1+\delta)\alpha} \gamma^\alpha (\log \gamma + (1+\delta) \log\log q)^{1-\alpha} \ .$$

Again ignoring the coefficients,

$$(\log q)^{(1+\delta)\alpha+\epsilon} (\log\log q)^{1-((1+\delta)\alpha+\epsilon)} \frac{(\log \gamma + (1+\delta)\log\log q)^{1-\alpha}}{(\log\log q)^{1-((1+\delta)\alpha+\epsilon)} (\log q)^\epsilon}$$

for any $\epsilon > 0$. We now note that as $q \to \infty$, the fractional term tends to 0. Hence we have that

$$(\log q^g)^\alpha (\log\log q^g)^{1-\alpha} \gg (\log q)^{(1+\delta)\alpha+\epsilon} (\log\log q)^{1-((1+\delta)\alpha+\epsilon)} \ .$$

This implies the running time is bounded above by $L_q[(1+\delta)\alpha + \epsilon, \beta' + o(1)]$ for any $\epsilon > 0$ and $\beta' \geqslant 0$. This time we note that if $\delta < 1$, and since $\alpha = 1/2$, we can choose ϵ so that $(1+\delta)\alpha + \epsilon < 1$.

It should not be construed that either of these two results imply the existence of a subexponential algorithm for the ECDLP. Examining them more deeply, they in fact suggest that our current index calculus approaches for solving the HCDLP are incapable of yielding algorithms with a running time in the range required for either theorem. To derive this conclusion we note that while the current algorithms for solving an instance of an HCDLP on a curve of genus g over $\mathbb{F}_q$ utilize factor bases which are subexponential in terms of q^g, they are exponential in terms of q. Hence, using such algorithms to solve an instance of the ECDLP by embedding it in the Jacobian of a hyperelliptic curve can not result in a subexponential algorithm. This does not exclude the possibility of such techniques being effective in constructing an exponential algorithm which has a better run-time than Pollard-rho.

Using the same techniques as above, we can make the following somewhat perverse observation.

Theorem 5. *If there exists a subexponential algorithm for hyperelliptic curves of genus 2, then there exists a subexponential algorithm for elliptic curves.*

The proof follows as above using $n = 2$ (in fact, the previous theorem is also true with 2 replaced by any other fixed genus that may be written in the form $\lfloor n + \frac{n-1}{2} \rfloor$). It is important to note that the resulting algorithm would have worse overall complexity and the size of the elliptic curve for which the asymptotics would assert themselves is undoubtedly very large, but it would still be subexponential. The conundrum that arises from this observation is that the converse statement is not necessarily true.

3.4 The Analogue between HCDLP and IQDLP

While solving the HCDLP and IQDLP problems appear to be linked only superficially since they are both discrete logarithm problems, the connection between them runs much deeper. If we consider the original algorithm developed by [1] to solve the HCDLP in high genus hyperelliptic curves, it has the same fundamental structure as the IQ-MPQS. That is, they both find relations by searching for elements with smooth norms in certain quadratic extensions. If we restrict our attention to the case of imaginary hyperelliptic curves (when the degree of f is odd), then we have the following diagram can be used to demonstrate the connection.

$$\begin{tikzcd}
K_\Delta \arrow[rr, dotted, no head] \arrow[dr, no head] \arrow[dd, no head, "2"'] & & \mathbf{K}(C) \arrow[dr, no head] \arrow[dd, no head, "2"] & \\
& \mathcal{O}_\Delta \arrow[rr, dotted, no head] \arrow[dd, no head, "2"] & & \mathbf{K}[C] \arrow[dd, no head, "2"] \\
\mathbb{Q} \arrow[rr, dotted, no head] \arrow[dr, no head] & & \mathbf{K}(x) \arrow[dr, no head] & \\
& \mathbb{Z} \arrow[rr, dotted, no head] & & \mathbf{K}[x]
\end{tikzcd}$$

In particular, solving the HCDLP in $\mathrm{Pic}^0_{\mathbf{K}}(C)$ is equivalent to solving the same problem in the *ideal class group* of $\mathbf{K}[C]$. Fundamentally, both the HCDLP and class group computations in imaginary quadratic orders are equivalent to solving the discrete logarithm problem in ideal class groups of a quadratic extension. Indeed, if we consider the algorithm presented in [6], it can be considered to solve the problem in both situations. Although there are some subtle differences that arise in the analysis of the complexity, they do not serve to effect the the exponent α which has the greatest impact on the asymptotic run-time.

The problem that prevents the development of a suitable IQ-NFS is the same problem that prevents the development of a better algorithm to solve HCDLP's. Namely, how does one find an extension of a quadratic extension which yields suitable ideal-homomorphisms. Considering the strong analogy between the two situations, it seems plausible that finding a solution in one of the two settings could easily be extended to the other.

4 Conclusion

We have presented some indication that the techniques of the number fields sieve may not be applicable to computations in imaginary quadratic number fields in a profitable way. In particular, it is unknown how to design fundamental core elements of an IQ-NFS algorithm, and even if this were known, it would not be clear whether or how such an algorithm could be useful (i.e. profitable). Moreover, we gave an outline how the existence of an IQ-NFS with the running time asymptotics $L\left[\frac{1}{3}\right]$ could conceivably be used to develop an algorithm to solve elliptic curve related computational problems with subexponential running time. It is worthwhile to point out that the analogy between these two settings is not restricted to algorithms of index-calculus type. It follows for example that if there is no subexponential algorithm to solve the ECDLP, then it is likely that $L_{|\Delta|}\left[\frac{1}{2}, c + o(1)\right]$ is the best achievable running time for the IQDLP.

As pointed out in the introduction, due to the somewhat speculative nature of this article, it is to be understood as a starting point for further research. For example, it would be interesting to establish rigorously the computational equivalence of the discrete logarithm problems for number field and function field class groups.

References

1. Adleman, L. M., DeMarrais, J., and Huang, M.-D. A subexponential algorithm for discrete logarithms over hyperelliptic curves of large genus over GF(q). *Theoretical Computer Science 226*, 1–2 (1999), 7–18.
2. Buhler, J. P., Lenstra, Jr., H. W., and Pomerance, C. Factoring integers with the number field sieve. In *The development of the number field sieve*, A. K. Lenstra and H. W. Lenstra, Eds., no. 1554 in LNM. Springer–Verlag, 1993, pp. 50–94.
3. Cohen, H. *A Course in Computational Algebraic Number Theory*, vol. 138 of *GTM*. Springer–Verlag, 1995.
4. Crandall, R., and Pomerance, C. *Prime Numbers: A Computational Perspective*. Springer–Verlag, 2000.
5. Damgård, I., and Fujisaki, E. A statistically-hiding integer commitment scheme based on groups with hidden order. In *Advances in Cryptology – ASIACRYPT 2002* (2002), Y. Zheng, Ed., vol. 2501 of *LNCS*, Springer-Verlag, pp. 125–142.
6. Enge, A., and Gaudry, P. A. A general framework for subexponential discrete logarithm algorithms. *Acta Arithmetica 102*, 1 (2002), 83–103.
7. Gordon, D. M. Discrete logarithms in $gf(p)$ using the number field sieve. *SIAM Journal of Discrete Mathematics 6*, 1 (1993), 124–138.
8. Hamdy, S. IQ cryptography: A secure and efficient alternative. *Journal of Cryptology* (2003). Submitted.
9. Jacobson, Jr., M. J. Applying sieving to the computation of quadratic class groups. *Mathematics of Computation 68*, 226 (1999), 859–867.
10. Jacobson, Jr., M. J. *Subexponential Class Group Computation in Quadratic Orders*. PhD thesis, Technische Universität Darmstadt, Fachbereich Informatik, Darmstadt, Germany, 1999.
11. Koblitz, N. *Algebraic Aspects of Cryptography*, vol. 3 of *Algorithms and Computation in Mathematics*. Springer–Verlag, 1998.
12. Schirokauer, O. Discrete logarithms and local units. *Philosophical Transactions of the Royal Society of London, Series A. 345*, 1676 (1993), 409–423.
13. Schirokauer, O. Using number fields to compute logarithms in finite fields. *Mathematics of Computation 69*, 231 (2000), 1267–1283.
14. Schirokauer, O., Weber, D., and Denny, T. Discrete logarithms: The effectiveness of the index calculus method. In *Algorithmic Number Theory, ANTS-II* (1996), H. Cohen, Ed., vol. 1122 of *LNCS*, Springer-Verlag, pp. 337–361.
15. Weber, D. Computing discrete logarithms with the general number field sieve. In *Algorithmic Number Theory, ANTS-II* (1996), H. Cohen, Ed., vol. 1122 of *LNCS*, Springer-Verlag, pp. 391–403.